MINAR International Journal of Applied Sciences and Technology
ISSN: 2717-8234
Article type: Research Article

Received: 16/03/2022 **Accepted:** 05/04/2022 **Published:** 01/06/2022

USING SOFTWARE ENGINEERING TO DESIGN AND IMPLEMENT MIXER OF MULTIPLE CAMERAS, MICROPHONE, AND SCREENS

Naktal Moaid EDAN[1]
University of Mosul, Iraq
Sanabil A MAHMOOD[2]
University of Mosul, Iraq

Abstract

To connect with user media devices such as microphones and cameras, browsers formerly required Flash. Flash has essentially replaced Web Real-Time Communication (WebRTC) in recent years. The principles of WebRTC have not agreed on how browsers can capture audio, video, and data or screens. The major purpose of this research is to create a new WebRTC recording method that uses Google Chrome and Firefox to capture a mixture of cameras, microphones, and screens. In addition, this study used Software Engineering (SE) concepts such as the software design process and interface design, which is the description of a system's relationships with its environment, including the analysis and development of the designed systems. The MultiStreamsMixer.js libraries were used to design and implement a mixer of many cameras, microphones, and screens using the Ethernet and Wireless of the 4th generation (4G) network. The suggested technique also makes use of the JavaScript Library to capture audio, video, and screen (two-dimensional and three-dimensional animations); as well as numerous audio and video codecs, such as VP8 and VP9 for video and Opus for audio, were also utilized in Chrome and Firefox. Additionally, multiple bitrates ranging from 100 bytes per second to 1 Gigabit per second were also tested. Besides, various resolutions ranging from 480p to HD (3840* 2160) and frame rates ranging from 5 to 70, increasing by 5% each time were applied. In addition, the recording device, Quality of Experience (QoE) over real operators, and resources were evaluated.

Keywords: Software Engineering (SE); Software Design Process; Web Real-Time Communication; Quality of Experience (QoE); 4th internet generation.

http://dx.doi.org/10.47832/2717-8234..118

[1] naktal.edan@uomosul.edu.iq, https://orcid.org/0000-0003-0799-1858
[2] sanabil_2000@uomosul.edu.iq, https://orcid.org/0000-0002-5939-5890

This article has been scanned by **iThenticat** No **plagiarism** detected

Introduction

1.1. Overview

Software Engineering (SE) is described as the procedure of evaluating the requirements of the user and then designing, developing, and testing software to meet those requirements. Thus, in this effort, an application of mixer cameras, microphones, and screens was designed and implemented successfully using MultiStreamsMixer.js libraries. So, using SE on a certain module allows you to change the software's appearance, functionality, and so on. Additionally, Web Real-Time Communication (WebRTC) was announced in 2011 by the Internet Engineering Task Force (IETF) and World Wide Web Consortium (W3C) [1][2][3][4][5]. WebRTC is a default option and a set of frameworks [6] that enable cooperating video and data exchanges [7][8][9][10]. It also has many advantages, including no costs, no certificate, without plug-ins, and it enables flexible software engineering and the identification of appropriate interactions by using the WebRTC approach [11][12][13]. In [14], highlighted that recording has modified the delivery and consumption of

education, Furthermore, WebRTC has offered various Application Programming Interfaces (APIs) for usage in recording, as specified in the relevant W3C documents that have been applied on modern webpages [15]. WebRTC misses high-end videoconferencing features like session recording [16]. Furthermore, [14] WebRTC standards, have not taken into account what is going on with computer recording (screen) and the recording of all media material. WebRTC analyzes the causes for all surfaces in demand to use online apps for broadcasting and capturing that is facilitated by the microphone, camera, and screen. The main goals of this study are as below:

A. Structure and exam a WebRTC recording mechanism.

B. Run the instrument via the 4G network.

C. Use many multimedia codecs.

D. Use numerous bitrates.

E. A variety of resolutions were investigated.

F. Arrangements rates were tested, including several bitrates.

G. An assessment of the recording mechanism and resources was achieved.

As a result, an innovative execution was achieved across multiple networks, browsers, codecs, peers, and MultiStreamsMixer.js libraries establishing multiple cameras, and microphones around the same time, retaining transmitting dynamic as long as the user requires, saving the record, and only using the recording as connecting with a complete screen. The plan of this research is as defined; Section II debates the research review. In section III, the performance of the procedure of the paper is clarified with execution and investigation. Section IV shares the estimation. Lastly, Section V is the decision and upcoming effort.

1.2. Research Review

In [17][16][14][18][19], indicated that the key WebRTC concerns are camera and microphones capture and screen; in particular, it has not been developed. Also, in [20] the author stated that an interface for screens using WebRTC for Automaton was created using the “getUserMedia API”, but the work was never proven or shown. In addition, [21] estimated that using “MediaRecorder API” can sustain recording on the Web. In contrast, [22] demonstrated that session recording via cameras or screens is a substantial difficulty because the WebRTC standard does not provide a streaming mechanism for gathering and storing information. Accordingly, [18][23] WebRTC necessitates several workarounds, such as recording features to permit machines to engage in restricted network scenarios and a WebRTC conferencing idea to aid in recording discussions. Further, [11] established that in the course of the test, a screen recorder is essential for information.

2. Approach, Implementation, and Analysis

2.1. Approach

For the construction and recording design and test of this software, various modules for the screen, and canvas (2nd and 3rd animation) were utilized. A task manager was also used to analyze CPU performance, as well as a NetCommWireless access point that can deliver 4G, resources, Google Chrome and Firefox were also used on the client-side. In addition, one PC is linked to the 4G networksvia (Ethernet and Wireless).

2.2. Implementation

Using the "MediaRecorder API", and WebRTC JavaScript code, a new system for recording has already been devised and implemented in this study for screens and "(2nd + 3rd animation) recording". To complete this task, this submission has evaluated software engineering levels, including design and development processes. It has also been presented under the following headings: Set up a major browser (Index HTML), use the "RecordRTC API", and use the "node.js server".

2.2.1. Set the main Page

Firstly allow access many cameras, microphones, and screens; MultiStreamsMixer.js libraries were used including the "getUserMedia API". As result, "RecordRTC" can start recording videos. Furthermore, JavaScript languages were printed to construct this presentation, as shown: (a) Begin/pause capturing; **(b)** Save the recording to a disk, **(c)** break the recording automatically within 4 minutes (to delete all the captured records), **(d)** Add extra media-streams (to existing recordings), **(e)** uses the following libraries:

1. MultiStreamsMixer.js libraries to offer.
    - A mixture of several cameras or videos.
    - A mixture of several microphones.
    - A mixture of audios, such as Wav and Mp3.
    - A mixture of many screens
2. "GetRecorderType.js".
3. "MRecordRTC.js".
4. "MediaStreamRecorder.js".
5. "StereoAudioRecorder.js".
6. "CanvasRecorder.js".
7. **"**RecordRTC.promises.js".
8. "WebAssemblyRecorder.js".
9. "MRecordRTC.js" (to bring multiple records into one place).

It adjusts the video width to 640 pixels wide and high to 480 pixels high, as illustrated in Figure (1). After a creator opens the browser, it displays an audio and video "MediaStream," which can be retrieved by capturing the screen using the "navigator.getUserMedia" method. Once a media has accessibility to the camera and microphone, it will begin streaming the multimedia and displaying it before saving it to a hard disk. The pseudocode for this experiment is shown in Figure (2). A user must close or reload the web page to leave the room or modify the setup, such as determination, bitrates, and codec. You may also control the camera or microphone's streaming, exploit the screen, or silence the video at any time.

Figure1. The principal browser via Chrome presents using different kind of cameras, and microphones and get mixed screen.

Figure2. Multiple cameras via Chrome using VP8 video codec.

- Adjust MultiStreamsMixer.js;
- Adjust JavaScript;
- Adjust Node.js;
- Adjust K1= Access local resources (Camera and Microphone);
- Adjust SS= Start Flooding;
- Adjust PT= Pause Flooding;
- Adjust Media Stream;
- SWITCH Twitch Recording;
- CASE1: open a new page;
- First: internal available;
- IF KI = yes; THEN access camera and microphone;
- ELSE connect the internal;
- First: start flooding;
- IF SS = yes; THEN stream and save media on disk;
- ELSE SS= end flooding and run recorded media;
- Second: pause flooding;
- IF PT = yes; THEN PT = end flooding and recording;
- ELSE PT = restart flooding;
- Third: change the situation;
- IF CR= yes; THEN modification resolution, frame rates, etc.;
- ELSE go out by default;
- End

Figure 3. Application pseudocode.

2.2.2. Produced Mechanism

RecordRTC API and JavaScript Techniques were utilized to construct and utilise this mechanism. Using Google Chrome, Firefox, and Opera, the "RecordRTC library" was used to establish and

establish an original meeting for screens. It has also provided a slew of new JavaScript methods for working with local and remote streams. Figure (1), shows an example of a software engineering development approach (3).

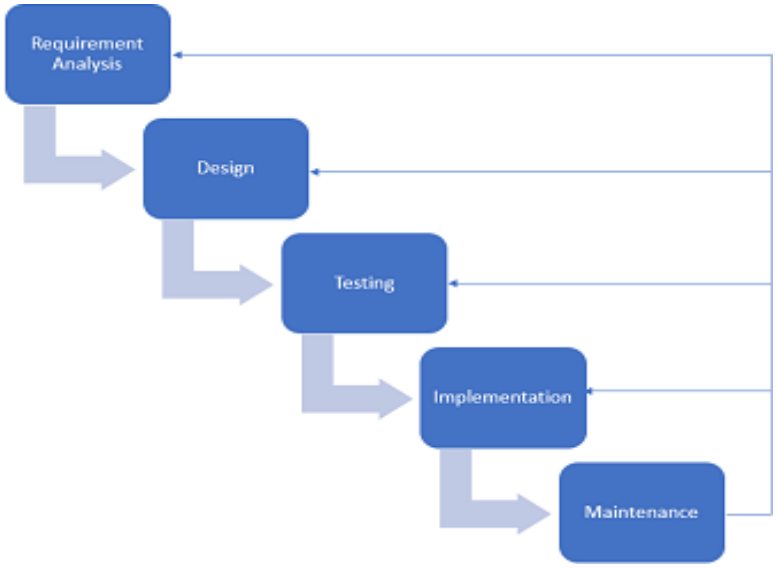

Figure4. the waterfall software development approach to present design steps.

2.3. Analysis

2.3.1. Signalling Protocol

This mechanism was studied independently for ten users to see how long it took them to become ready. This was constructed based on a net analysis of Google Chrome and Firefox, elements during actual communication. The overall time was deliberate, and it takes 122 milliseconds (ms) to get prepared and 455 milliseconds (ms) to start streaming media. This system may concurrently establish, and transmit screens. The postponement varied slightly depending on whether you use Chrome and Firefox. The performance of audio, films, and the screen, on the other hand, was unaffected by CPU load or bandwidth use; in particular, media streaming between internal devices was unaffected.

2.3.2. Quality of Video Conferencing

Individual tests were conducted among 8 users across the 4G networks to assess audio, video, and screen quality. As a result, the audio, video, and screen quality were all great. As a consequence, as demonstrated in Table (1), employing the forming method for recording audio, videos, and screens is effective.

Table 1, the value of the screen among users over (LAN & WAN) the 4G networks.

No.	**Browser**	**OS**	**Type**	**Codec**		**Period**	**Value of Audio and video**	**Value of Screen**
				Video	**Audio**			
1.	Chrome	Win 10	Multimedia and Screen	VP8, VP9	OPUS	1 - 10 minute	Excellent	Excellent
2.	Firefox	Win 10	Multimedia and Screen	VP8, VP9	OPUS	1 - 10 minute	Excellent	Excellent

2.1.3. Quality of Experience (QoE)

The author indicated that QoE is important and has been recognized as a particular framework in broadcasting communication and that the ITU-T group has adopted it [24]. Users participated in this test by filling out a questionnaire to provide feedback on their actual customer knowledge, as shown in Table (2). This presentation demonstrated great screen recording quality, particularly amongst 10 users across the 4th G networks.

Table 2, QoE of 15 customers via 4G network.

Questions	Very Bad	Bad	Fair	Good	Excellent
Access the system via the RecordRTC					15
Assess the system at all				1	14
Assess the quality of multimedia through the meeting			2	10	3
Assess the quality of the screen throughout the meeting				1	14
Assess the easiness of the usage			1	4	10
Is this system convince you to use RecordRTC				5	10

4. Evaluation

It was demonstrated that the proposed program can be used to maintain recording across a variety of browsers, including Google Chrome and Firefox. To start, stop, and include media streaming via the 4G networks, this solution employs a new WebRTC recording method. Furthermore, it provides video conferencing, maintains media streaming efficiency, and manages self-streams. It was made without the use of any external devices or a commercial cloud/server. This project is the first to use the 4G network to develop a WebRTC recording appliance for screen recording. The Safari browser, on the other hand, is not supported.

5. Decision and Upcoming Work

A massive effort was considered in this research that started from linking between WebRTC and SE until finding the way that can enable and support the communication among many users to present a mixer of cameras, microphones and screens. Thus, a new WebRTC technique was built and tested in this research using the 4th generation network. The MultiStreamsMixer.js libraries were utilised to start, stop, and restart cameras, microphones, and screens for streaming. This result is powerful because it provides a visual demonstration of true face-to-face communication across various networks and browsers. Furthermore, it improves communication, connections, and efficiency between customers and teams. This endeavour will be expanded in the future to include additional scalable cameras, microphones and screens.

References

1- B. Y. Julian. Jang-Jaccard, Surya. Nepal, Branko. Celler (2016), "WebRTC-based video conferencing service for telehealth," Computing, vol. 98, no. 1–2, pp. 169–193.
2- N. Edan, A. Al-Sherbaz, and S. Turner (2018), "Design and implement a hybrid WebRTC signalling mechanism for unidirectional & bi-directional video conferencing," Int. J. Electr. Comput. Eng., vol. 8, no. 1.
3- N. M. Edan, A. Al-Sheraz, and S. Turner (2017), "WebNSM: A Novel WebRTC Signalling Mechanism for One-to-Many Bi-directional Video Conferencing," in Proceedings of 2018 SAI Computing Conference, pp. 1–6.
4- S. Perreault (2010(, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations," USA.
5- V. P. I. Baz Castillo, J. Millan Villegas (2014), "The WebSocket Protocol as a Transport for the Session Initiation Protocol (SIP)," Spain.
6- M. Phankokkruad and P. Jaturawat (2015), "An Evaluation of Technical Study and Performance for Real- Time Face Detection Using Web Real-Time Communication," in International Conference on Computer, Communication, and Control Technology (I4CT), no. I4, pp. 162–166.
7- M. L. Giuliana. Carullo, Marco. Tambasco, Mario. Di Mauro (2016), "A Performance Evaluation of WebRTC over LTE," in 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS), pp. 170–175.
8- N. M. Edan, A. Al-sherbaz, and S. Turner (2017), "Design and Evaluation of Browser-to-Browser Video Conferencing in aWebRTC," in 2017 Global Information Infrastructure and Networking Symposium (GIIS), p. 4.
9- S. Owesen-Lein (2015), "Unified Communication and WebRTC," Norwegian University of Science and Technology.
10- C. J. S. Nandakumar (2018), "Annotated Example SDP for WebRTC: draft-IETF-rtcweb-SDP-09," USA.
11- L. O. D. N. Eirik. Fosser (2016), "Quality of Experience of WebRTC based video communication," Norwegian University of Science and Technology.
12- N. M. Edan, A. Al-sherbaz, and S. Turner (2017), "WebNSM : A Novel Scalable WebRTC Signalling Mechanism for Many-to-Many Video Conferencing," in 3rd IEEE International Conference on Collaboration and Internet Computing (CIC), vol. 2, pp. 1–7.
13- T. F. Michael. Adeyeye, Member, Ishmeal. Makitla (2013), "Determining the signalling overhead of two common WebRTC methods: JSON via XMLHttpRequest and SIP over WebSocket," in Africon, Pointe- Aux-PimentsConference, pp. 0–4.
14- S. Skrødal (2016), "SA8T2 Internal Deliverable Technology Scout : Stream and record lectures with WebRTC".
15- N. PINIKAS (2016), "A Webrtc Based Platform for Synchronous Online Collaboration and Screen Casting," Technological Educational Institute of Crete.
16- M. Pasha, F. Shahzad, and A. Ahmad (2016), "Analysis of challenges faced by WebRTC videoconferencing and a remedial architecture," Int. J. Comput. Sci. Inf. Secur., vol. 14, no. 10, pp. 698– 705.
17- B. Bos, E. Davies, L. Desmet, S. Farrell, M. Johns, and R. Wenning (2014). "Strategic Research Roadmap for European Web Security".
18- M. Walter (2015). "WebRTC multipoint conferencing with recording using a Media Server," Stuttgart Media University.
19- J. Rodriguez, Pedro. Cerviño Arriba, Javier. Trajkovska, Irena. Salvachua (2019). "Advanced videoconferencing based on webrtc," in IADIS Multi Conference on Computer Science and Information Systems, p. 6.
20- P. Kinlan, 2016 "Screen recording on Android with getUserMedia and WebRTC," [online] available from https://medium.com/dev-channel/screen-recording-on-android-with-getusermedia-and-webrtc- c32ba9d29c28. [Accessed November 25, 2019].
21- S. Penadés, 2016 "Record almost everything in the browser with MediaRecorder," [online] available from https://hacks.mozilla.org/2016/04/record-almost-everything-in-the-browser-with-mediarecorder/.[Accessed October 7, 2019].
22- P. Rodríguez, J. Cerviño, I. Trajkovska, and J. Salvachúa (2012). "Advanced Videoconferencing Services Based on WebRTC," in IADIS International Conferences

Web Based Communities and Social Media 2012 and Collaborative Technologies, pp. 180–184.
23- B. García, L. López-Fernández, F. Gortázar, and M. Gallego (2019). “Practical evaluation of VMAF perceptual video quality for webRTC applications,” Electron., vol. 8, no. 8, pp. 1–15.
24- R. C. Streijl, S. Winkler, and D. S. Hands (2016). “Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives,” Multimed. Syst., vol. 22, no. 2, pp. 213–227.